**Selenium Grid on AWS EKS**

**By: Kavan Dalwadi**

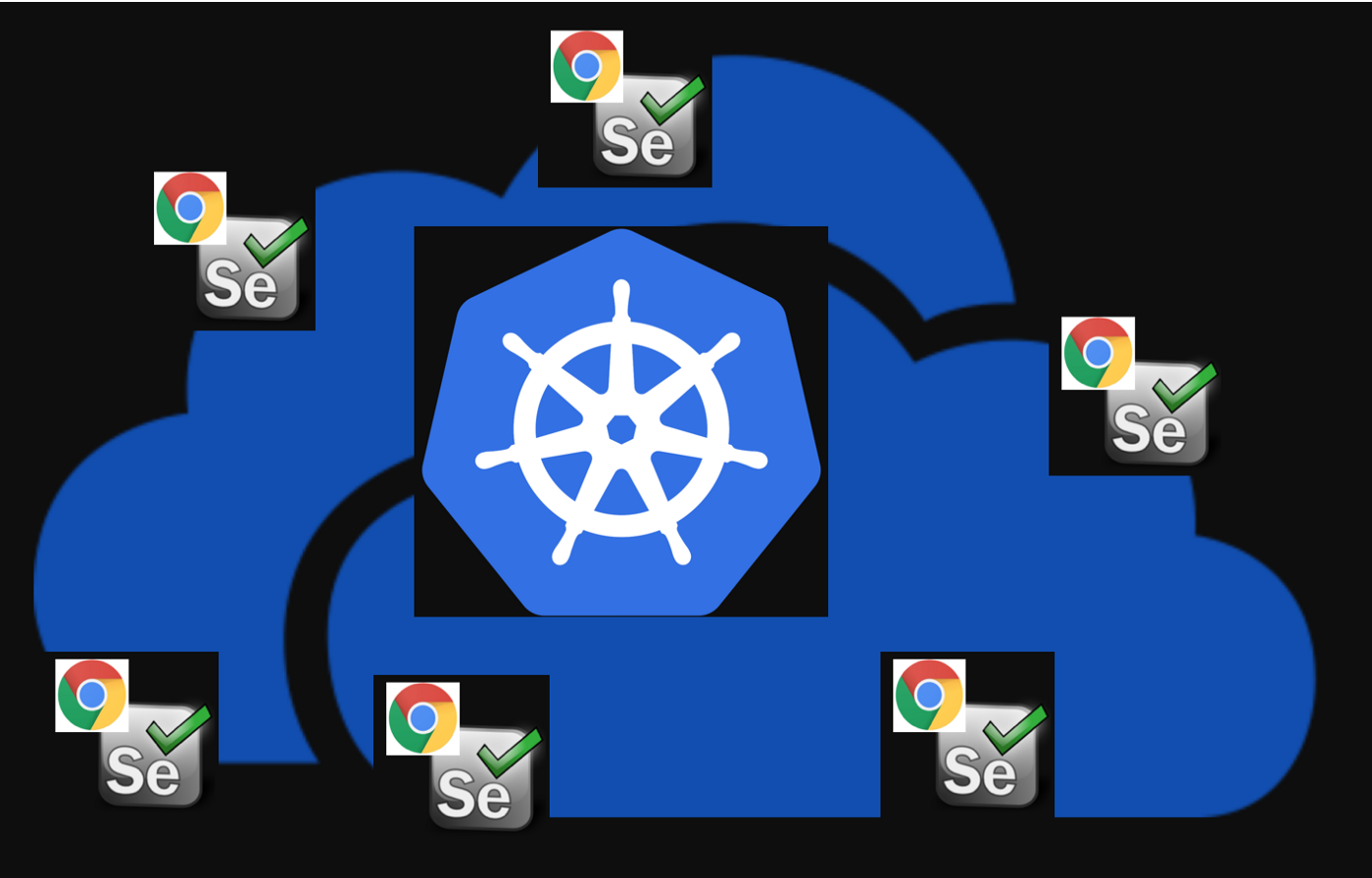
# TOPICS -

## Create K8s Cluster in AWS.

## Selenium Grid 4 and its architecture.

## Deploy Selenium Grid in cluster.

## Run a Selenium Job.



# Create AWS EKS cluster

## Prerequisites

Install AWS CLI

<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>

Install eksctl

<https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html>

Install kubectl

<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

Create and configure AWS EKS cluster

We can create aws EKS cluster using multiple ways:

1. Using console
2. Using terraform
3. Using eksctl cli

We are going to use eksctl to create our EKS cluster.

First we need to configure aws cli, for that use command:

aws configure

To create EKS cluster run command:

eksctl create cluster --name <name> \

--version 1.22 \

--region <region> \

--nodegroup-name my-nodes \

--node-type t3.small \

--managed --nodes 2 \

--ssh-access \

--ssh-public-key <key\_name> \

--node-ami-family Ubuntu2004

This will spin up an AWS EKS cluster with 2 nodes, in a specified region and having specified name.

Version flag will use 1.22 k8s version, also you can use 1.19, 1.20, 1.21 version.

We are also specifying node type t3.small, you can change as per your requirements.

To enable ssh use flag --ssh-access and specify key using --ssh-public-key, also you can change AMI of nodes from ubuntu to amazon linux.

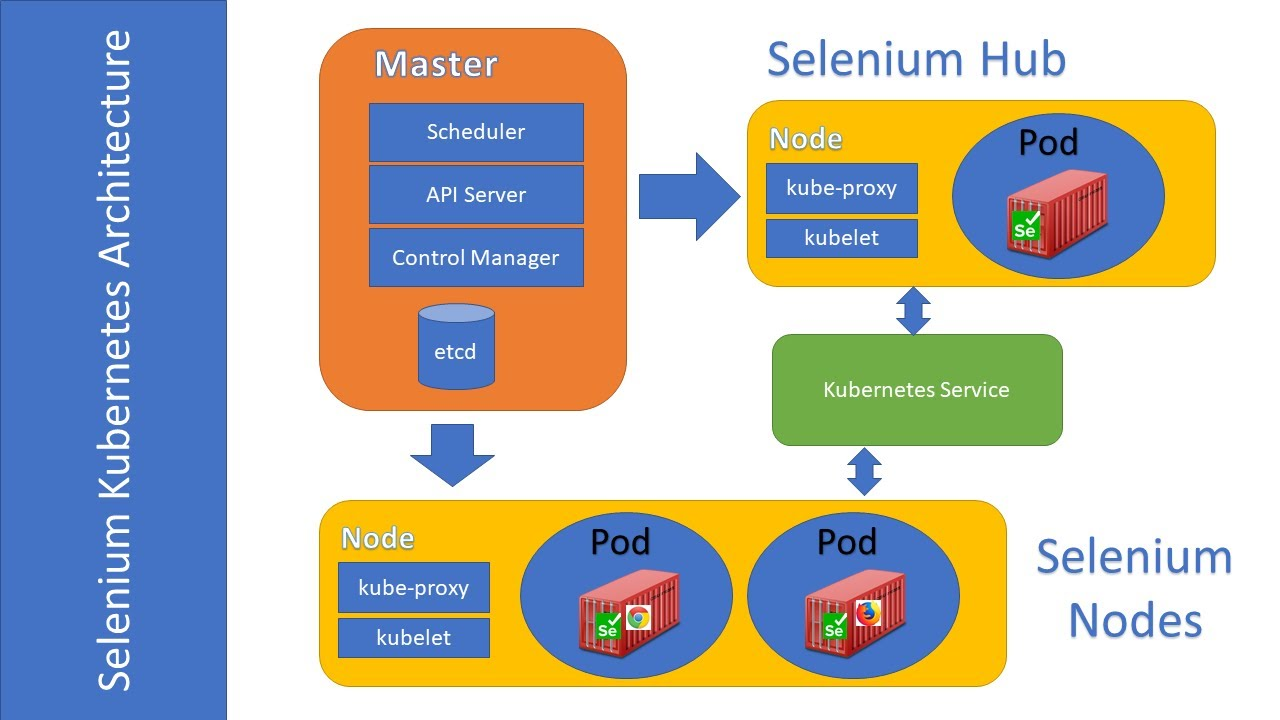
For **Debugging** use command, you can find various flags and their use case -

eksctl create cluster --help

## What is Selenium Grid 4 -

| **Older Selenium Grid** | **Selenium Grid 4** |
| --- | --- |
| **Has one mode:**  Hub and Node | **Has 3 Modes:**  Standalone  Hub and Node  Distributed |
| **Has two components:**  Hub, Node | **Has Six components:**  Router  Distributer  Nodes  Session Queue  Session Map  Event Bus |

## Hub and Node Architecture -



# 

# DEPLOYING SELENIUM GRID in EKS

# Using manifest files -

<https://github.com/kubernetes/examples/tree/master/staging/selenium>

kubectl apply -f <file\_-name>

# Using Helm Charts -

<https://github.com/SeleniumHQ/docker-selenium/tree/trunk/charts/selenium-grid>

1. helm repo add docker-selenium https://www.selenium.dev/docker-selenium
2. helm repo update
3. helm search repo docker-selenium --versions
4. helm install selenium-grid docker-selenium/selenium-grid

**Or install full grid (Router, Distributor, EventBus, SessionMap and SessionQueue components separated)**

1. helm install selenium-grid docker-selenium/selenium-grid --set isolateComponents=true

By default selenium-hub service will be of type ClusterIP, to access the selenium UI,

You can change service type to NodePort -

kubectl patch svc <service\_name> --type='json' -p '[{"op":"replace","path":"/spec/type","value":"NodePort"}]'

To run a test in python, in the EKS cluster we can create a pod with python image and run our test file in that pod, to create a pod use command:

kubectl run selenium-python --tty -i --image=python:slim bash

After pod is created, if you want to execute any test file in it, use command:

kubectl exec -it selenium-python -- bash

To view Selenium hub UI run following command:

kubectl get node -o wide

kubectl get svc/selenium-hub

After getting the output you can visit

http://<node\_IP>:<node\_Port>

More about Selenium with Python:

<https://www.browserstack.com/guide/python-selenium-to-run-web-automation-test>  
  
Sample Test Code:

from selenium import webdriver

def check\_browser(browser):

if browser == "CHROME":

options = webdriver.ChromeOptions()

elif browser == "FIREFOX":

options = webdriver.FirefoxOptions()

driver = webdriver.Remote(

command\_executor='http://selenium-hub:4444/wd/hub',

options=options

)

driver.get("http://www.google.com")

assert "google" in driver.page\_source

driver.quit()

print("Browser %s checks out!" % browser)

check\_browser("FIREFOX")

check\_browser("CHROME")